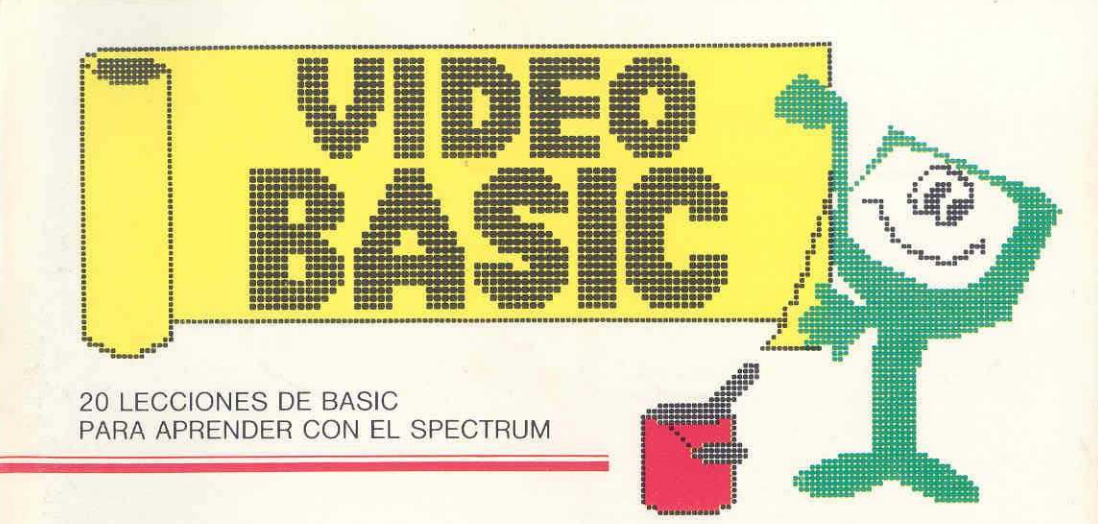
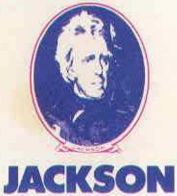


20 LECCIONES DE BASIC
PARA APRENDER CON EL SPECTRUM

INGELEK



JACKSON

- Cómo funciona el teclado
- El código ASCII
- El juego de caracteres del Spectrum
- CODE, CHR$
- INKEY$
- FOR, TO, STEP, NEXT
- Los ciclos automáticos
- Videoejercicios
- Videojuego N. 5

5

Spectrum

16K/48K/PLUS

VIDEO BASIC
Una publicación de
INGELEK JACKSON
Director editor por INGELEK:
Antonio M. Ferrer
Director editor por JACKSON HISPANIA:
Lorenzo Bertagnolio
Director de producción:
Vicente Robles
Autor: Softidea
Redacción software italiano:
Francesço Franceschini,
Stefano Cremonesi
Redacción software castellano:
Fernando López, Antonio Carvajal,
Alberto Caffarato, Pilar Manzanera
Diseño gráfico:
Studio Nuovaidea
Ilustraciones:
Cinzia Ferrari, Silvano Scolari,
Equipo Galata
Ediciones INGELEK, S. A.
Dirección, redacción y administración,
números atrasados y suscripciones:
Avda. Alfonso XIII, 141
28016 Madrid. Tel. 2505820
Fotocomposición: Espacio y Punto, S. A.
Imprime: Gráficas Reunidas, S. A.,

ISBN del tomo 2: 84-85831-17-9
ISBN del fascículo: 84-85831-11-X
ISBN de la obra completa: 84-85831-10-1
Depósito Legal: M-15076-1985
Plan general de la obra:
20 fascículos y 20 casetes, de aparición quincenal,
coleccionables en 5 estuches.
Distribución en España:
COEDIS, S. A.
Valencia, 245. 08007 Barcelona.
INGELEK JACKSON garantiza la publicación de todos
los fascículos y casetes que componen esta obra y el
suministro de cualquier número atrasado o estuche
mientras dure la publicación y hasta un año después de
terminada.
El editor se reserva el derecho de modificar
el precio de venta del fascículo,
en el transcurso de la obra, si las circunstancias del
mercado así lo exigen.
Julio, 1985.
Impreso en España.



SUMARIO

Introducción

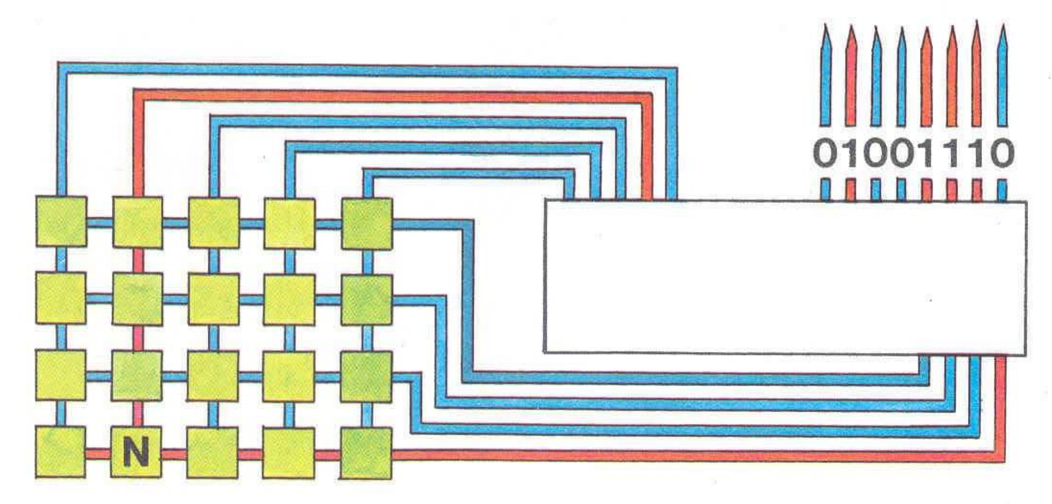
En esta lección profundizaremos en nuestro conocimiento del teclado que es el dispositivo principal para la entrada de datos.
Pero no todos los teclados son iguales y sus mecanismos y características varían de un tipo a otro, así como su principio de funcionamiento.
Ligados indisolublemente al teclado, veremos también el código ASCII y el juego de caracteres.
Después, conoceremos y aprenderemos a usar: CODE, CHR$, INKEY$ y FOR-TO-STEP-NEXT, lo que te permitirá repetir todas las veces que lo desees un grupo de instrucciones.
Y para terminar, una técnica indispensable para el programador: los ciclos controlados.

Funcionamiento y esquema de los tipos de teclado

El teclado constituye el dispositivo principal de entrada de informaciones con el que está dotado tu ordenador.
El teclado es el elemento que te permite comunicar todas las instrucciones y los datos que pretendes ejecutar o memorizar a la unidad central.
Un ordenador sin teclado sería como un coche sin volante: podrías ponerlo en marcha o pararlo, pero no lo podrías controlar ni utilizar.
Así que es importante que entiendas, más allá del simple y habitual uso diario, la estructura y los principios de funcionamiento del teclado de un ordenador.
Pero antes de tocar este tema es necesario precisar y aclarar con exactitud qué es lo que se entiende por el término «teclado». Esta palabra se refiere única y exclusivamente al dispositivo empleado para la entrada de datos.
Llamar «teclado» a todo un ordenador (como hacen algunas personas no demasiado informadas) es incorrecto y está absolutamente equivocado.
Como ya habrás visto, el teclado de tu Spectrum es fundamentalmente idéntico, tanto en su aspecto como en su funcionamiento, al de una simple máquina de escribir; también aquí será suficiente con pulsar la tecla correspondiente al carácter elegido.
En algunas ocasiones, con la combinación de dos o tres teclas pulsadas al mismo tiempo, se pueden obtener otras letras, símbolos o instrucciones que no están normalmente disponibles en las máquinas de escribir.
Un detalle, que quizá se te haya escapado, es el de la colocación de las teclas: están ordenadas según el estándar norteamericano llamado QWERTY. Este nombre, asignado a los teclados de tipo norteamericano, proviene de la colocación de las primeras seis teclas alfabéticas de la segunda fila.
En cambio, en los teclados llamados europeos, la Z ocupa la segunda posición en lugar de la W; de aquí el nombre de QZERTY.
Otras diferencias son la posición de la M y la disposición de la práctica totalidad de los símbolos y signos de puntuación.
De cualquier manera, en la práctica nada cambia; ambos teclados (norteamericano y europeo) se comportan de forma absolutamente idéntica y fiable desde

el punto de vista del funcionamiento.
Ahora intentaremos comprender cómo funciona en realidad un teclado, es decir, qué es lo que ocurre cuando pulsas una tecla de tu Spectrum.
Todas las teclas existentes en el teclado están conectadas eléctricamente (es decir, a traves de cables eléctricos) a un circuito integrado especial, que comprueba cúal de las teclas ha sido pulsada, y que emite un único y distinto código numérico binario de 8 bits para cada una de las teclas.
Así, la unidad central, cuando recibe una de estas combinaciones, es inmediatamente capaz (gracias a un programa memorizado en la ROM o en otro circuito electrónico) de distinguir y localizar el carácter pulsado, para poder así realizar las operaciones requeridas.
Por ejemplo, cuando pulsas la letra A, en la salida del circuito codificador (este es el término técnico empleado para nombrar este componente) aparece el código binario 01000001, cuyo correspondiente número decimal es el 65. A este código , y únicamente a él, le corresponde en la CPU el carácter A; así que no queda ninguna posibilidad de que surjan en la máquina errores y confusiones.

Un circuito integrado reconoce y localiza qué tecla ha sido pulsada y emite su código binario correspondiente.



El código ASCII

Los códigos numéricos que se asignan a cada uno de los caracteres de uso más común no se eligen arbitrariamente por parte de la marca fabricante, sino que son el fruto de la cooperación entre usuarios de aparatos e industrias asentadas en el campo de la elaboración de datos. En principio, tales códigos fueron elegidos con el objetivo de simplificar y estandarizar las comunicaciones entre diversos ordenadores, eliminando así todos los problemas relacionados con las diferentes representaciones de datos e informaciones. La difusión cada vez más amplia de los

Decimal	ASCII	Decimal	ASCII	Decimal	ASCII
0	NUL	43	+	86	V
1	SOH	44	,	87	W
2	STX	45	–	88	X
3	ETX	46	.	89	Y
4	EOT	47	/	90	Z
5	ENQ	48	0	91	[
6	ACK	49	1	92	\
7	BEL	50	2	93	]
8	BS	51	3	94	∧
9	HT	52	4	95	—
10	LF	53	5	96	`
11	VT	54	6	97	a
12	FF	55	7	98	b
13	CR	56	8	99	c
14	SO	57	9	100	d
15	SI	58	:	101	e
16	DLE	59	;	102	f
17	DC1	60	<	103	g
18	DC2	61	=	104	h
19	DC3	62	>	105	i
20	DC4	63	?	106	j
21	NAK	64	@	107	k
22	SYN	65	A	108	l
23	ETB	66	B	109	m
24	CAN	67	C	110	n
25	EM	68	D	111	o
26	SUB	69	E	112	p
27	ESC	70	F	113	q
28	FS	71	G	114	r
29	GS	72	H	115	s
30	RS	73	I	116	t
31	US	74	J	117	u
32	espacio	75	K	118	v
33	!	76	L	119	w
34	"	77	M	120	x
35	#	78	N	121	y
36	$	79	O	122	z
37	%	80	P	123	{
38	&	81	Q	124	|
39	'	82	R	125	}
40	(	83	S	126	~
41	)	84	T	127	DEL
42	*	85	U		

ordenadores personales ha conseguido que dicha codificación, llamada ASCII (abreviatura de American Standard Codes for Information Interchange, es decir, Códigos Estándar Americanos para el Intercambio de Informaciones), se haya convertido de hecho en el «estándar» empleado en la totalidad de los ordenadores. Por lo tanto, a los caracteres alfabéticos y de puntuación presentes en tu Spectrum, les corresponden las mismas combinaciones de bits codificadas en los demás ordenadores, es decir, todos obtienen idénticos códigos con las mismas letras.

Teclas y teclados

El teclado está sometido a un constante trabajo mecánico. Su vida y duración dependen en gran parte de la calidad de las teclas; ésta puede oscilar entre algunas decenas de miles de pulsaciones, en los teclados de inferior calidad, y muchas decenas de millones. Veamos brevemente los distintos tipos de teclas, empezando por las mejores:

- **Teclas de efecto Hall:** Aprovechan la acción de un imán móvil sobre la corriente que atraviesa un semiconductor. Dotados de escasa mecánica, su vida media se mide en billones de pulsaciones.
- **Teclas capacitativas:** Se trata de condensadores cuya capacidad varía con la pulsación de la tecla. Tienen una vida de muchas decenas de millones de pulsaciones y se emplean en los mejores ordenadores personales.
- **Teclas reed:** Un contacto situado en el interior de una burbuja de vidrio (el contacto reed) se cierra mediante un pequeño imán montado en la cinta. La vida es de algunas decenas de millones de pulsaciones. A causa de la competencia de los teclados capacitativos, que son más duros, su uso se ha reducido mucho durante los últimos años.
- **Teclas mecánicas estándar:** Se emplean en muchos ordenadores personales; su vida depende de su calidad de fabricación. En el mejor de los casos su vida es de algunas decenas de millones de pulsaciones. Son sensibles a las condiciones ambientales: humedad, polvo.
- **Teclas de bola:** Son las empleadas en las calculadoras de bolsillo. Una bola de metal se vuelca por la presión de la tecla. Por lo general, su vida es limitada.
- **Teclados de membrana o película:** Las teclas están constituidas por dos

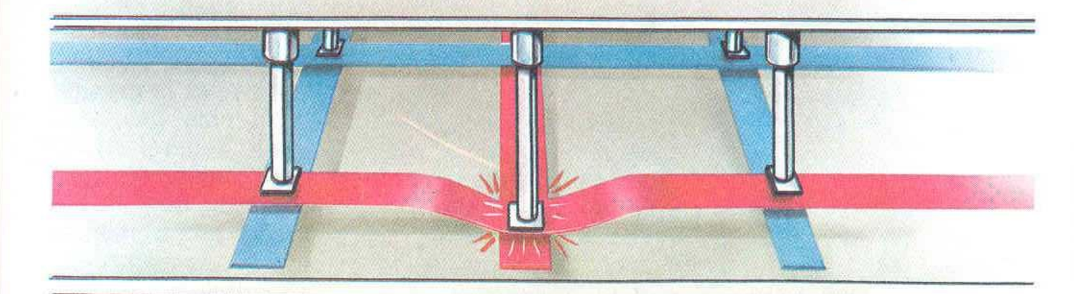
películas conductoras, tensadas y separadas por una lámina aislante perforada en la forma adecuada. Pulsando la película superior las dos hojas se tocan, cerrando el contacto. Normalmente, los teclados de membrana se emplean en los ordenadores económicos o para aplicaciones industriales, puesto que garantizan un excelente aislamiento de aquellos factores externos que pueden perjudicar su funcionamiento. Su duración depende de la calidad de fabricación.

Dejando aparte las diferencias de construcción, podemos explicar en una única secuencia las operaciones que ejecuta tu ordenador, o mejor dicho, su CPU, para analizar el teclado y localizar la tecla pulsada.

Veamos, de forma muy simplificada, qué es lo que ocurre:

1. Periódicamente, y en intervalos establecidos, la CPU interrumpe lo que está haciendo para prestar atención al teclado.
2. Ejecuta la llamada rutina de interrupción, que está memorizada en la ROM.
3. Controla (en la jerga, «scanning») la





situación del teclado, es decir, si alguna tecla ha sido pulsada.

4. Si se ha pulsado alguna tecla, la CPU obtiene la posición de dicha tecla.
5. Una vez verificado el punto 4, vuelve a comprobarlo un instante después (recuerda que la velocidad de la CPU es del orden del Megahertzio, es decir, de millones de veces por segundo), para poder excluir posibles interferecias.
6. Ahora la CPU vuelve a su trabajo, para reemprender más adelante la secuencia ya descrita.

Para concluir: el teclado es tu principal medio de comunicación con el ordenador; trátalo con cuidado.
Al ser un instrumento mecánico está sujeto a desgaste, y la duración de su vida depende del uso que hagas de él. Evita por lo tanto golpes bruscos, presiones excesivas de las teclas y sobre todo, cuando un programa no funcione... evita desahogar tu rabia en el teclado.

El juego de caracteres

Como bien sabes (perdona la pedantería, pero viene a cuento), tu Spectrum, como cualquier otro ordenador, conoce, sabe usar y memoriza únicamente números y además solamente binarios.
¿Cómo consigue entonces comprender y visualizar en pantalla la interrogación (?), el rótulo «adios», el asterisco (*)?
Es muy sencillo. Tu Spectrum convierte en números todos aquellos caracteres alfabéticos, numéricos y especiales (el juego de caracteres de la máquina) que es capaz de gestionar, enviar y recibir.

Para efectuar esta codificación emplea un código muy parecido al usado por todos los demás ordenadores personales: el ya citado código ASCII.
Por esta razón, tu ordenador también sabe «manipular» cadenas; las interpreta como una sucesión de números codificados que se corresponden con una tabla de caracteres presente en memoria.
Lo importante es que a cada código le corresponde un único carácter, para que así el ordenador nunca encuentre ambigüedades de interpretación.
Al igual que el ASCII, también el juego de caracteres de tu Spectrum es un código de 7 bits; por lo que son posibles 128 (es decir, 2^7) combinaciones, rápidas y fácilmente reconocibles con una primera ojeada, siendo de uso frecuente para cualquiera: letras alfabéticas, símbolos de puntuación, cifras numéricas.
Otros, en cambio, antes de asumir algún significado, requieren un instante de reflexión;

y aún otros, resultan absolutamente ajenos a los símbolos empleados normalmente por el hombre. Estos son los llamados caracteres especiales (o caracteres de control); a través de ellos puedes impartir órdenes especiales, que se adaptan a las características de funcionamiento de la máquina.
Aún careciendo de correspondencia con el lenguaje humano, los caracteres de control son importantísimos para tu Spectrum; gracias a ellos puedes, por ejemplo, indicar el final de una línea de programas (con la tecla ENTER), desplazar el cursor a cualquier lugar

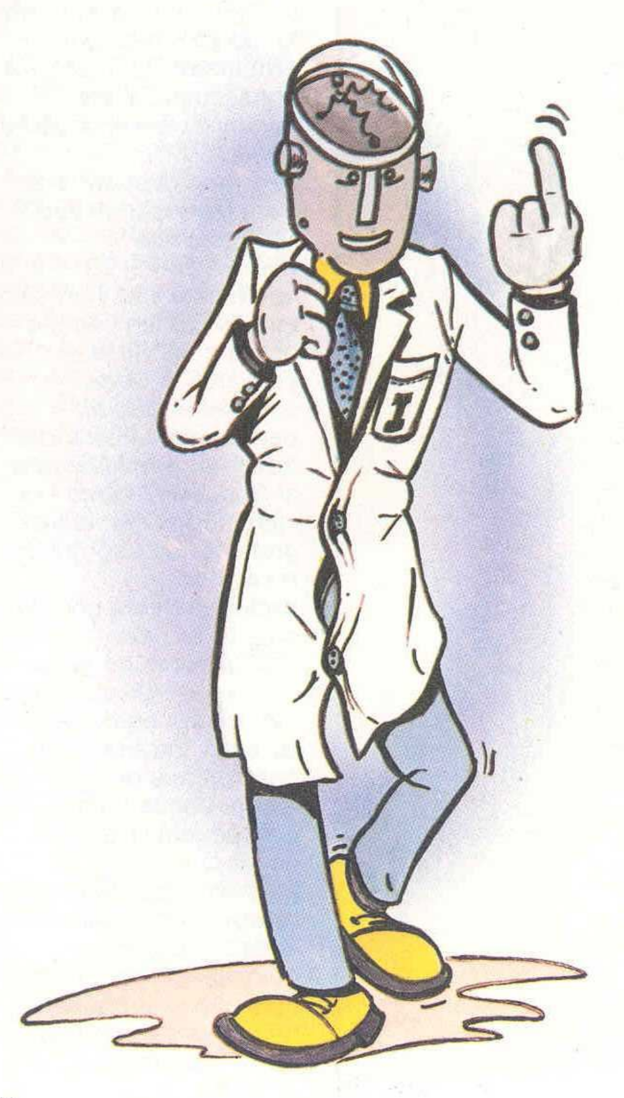
Código	*Carácter*	*Código*	*Carácter*	*Código*	*Carácter*
0		44	,	91	[
1		45	—	92	/
2		46	.	93	]
3	no empleados	47	/	94	↑
4		48	0	95	_
5		49	1	96	£
6	coma del **PRINT**	50	2	97	a
7	**EDIT**	51	3	98	b
8	cursor izquierda	52	4	99	c
9	cursor derecha	53	5	100	d
10	cursor abajo	54	6	101	e
11	cursor arriba	55	7	102	f
12	**DELETE**	56	8	103	g
13	**ENTER**	57	9	104	h
14	numero	58	:	105	i
15	no empleado	59	;	106	j
16	car. control **INK**	60	<	107	k
17	car. control **PAPER**	61	=	108	l
18	car. control **FLASH**	62	>	109	m
19	car. control	63	?	110	n
	BRIGHT	64	@	111	o
20	car. control	65	A	112	p
	INVERSE	66	B	113	q
21	car. control	67	C	114	r
	INVERSE	68	D	115	s
22	car. control **OVER**	69	E	116	t
23	car. control **AT**	70	F	117	u
24	car. control **TAB**	71	G	118	v
25		72	H	119	w
26		73	I	120	x
27	no empleados	74	J	121	y
28		75	K	122	z
29		76	L	123	{
30		77	M	124	|
31		78	N	125	}
32	espacio	79	O	126	~
33	!	80	P	127	©
34	"	81	Q	128	□
35	#	82	R	129	▝
36	$	83	S	130	▘
37	%	84	T	131	▀
38	&	85	U	132	▗
39	'	86	V	133	▐
40	(	87	W	134	▚
41	)	88	X	135	▜
42	*	89	Y	136	▖
43	+	90	Z	137	▞

Código	Carácter		Código	Carácter	Código	Carácter
138	▌		185	EXP	232	CONTINUE
139	▛		186	INT	233	DIM
140	▄		187	SQR	234	REM
141	▟		188	SGN	235	FOR
142	▙		189	ABS	236	GO TO
143	█		190	PEEK	237	GO SUB
144	(a)	caracteres gráficos definidos por el usuario	191	IN	238	INPUT
145	(b)		192	USR	239	LOAD
146	(c)		193	STR$	240	LIST
147	(d)		194	CHR$	241	LET
148	(e)		195	NOT	242	PAUSE
149	(f)	Caracteres gráficos definidos por el usuario	196	BIN	243	NEXT
150	(g)		197	OR	244	POKE
151	(h)		198	AND	245	PRINT
152	(i)		199	<=	246	PLOT
153	(j)		200	>=	247	RUN
154	(k)		201	<>	248	SAVE
155	(l)		202	LINE	249	RANDOMIZE
156	(m)		203	THEN	250	IF
157	(n)		204	TO	251	CLS
158	(o)		205	STEP	252	DRAW
159	(p)		206	DEF FN	253	CLEAR
160	(q)		207	CAT	254	RETURN
161	(r)		208	FORMAT	255	COPY
162	(s)		209	MOVE		
163	(t)		210	ERASE		
164	(u)		211	OPEN#		
165	RND		212	CLOSE#		
166	INKEY$		213	MERGE		
167	PI		214	VERIFY		
168	FN		215	BEEP		
169	POINT		216	CIRCLE		
170	SCREEN$		217	INK		
171	ATTR		218	PAPER		
172	AT		219	FLASH		
173	TAB		220	BRIGHT		
174	VAL$		221	INVERSE		
175	CODE		222	OVER		
176	VAL		223	OUT		
177	LEN		224	LPRINT		
178	SIN		225	LLIST		
179	COS		226	STOP		
180	TAN		227	READ		
181	ASN		228	DATA		
182	ACS		229	RESTORE		
183	ATN		230	NEW		
184	LN		231	BORDER		

de la pantalla (con las flechas ←,→), o borrar un carácter de la pantalla (DELETE).
Hemos dicho que el juego de caracteres de tu Spectrum emplea códigos de 7 bits, pero tu Spectrum es un ordenador de 8 bits; en consecuencia, las combinaciones posibles son 256 (2^8).
¿Por qué desperdiciar semejante capacidad? Las restantes $256 - 128 = 128$ combinaciones han sido empleadas en cambio por la casa Sinclair para definir otros caracteres que no pertenecen al estándar ASCII, sino únicamente al Spectrum, como por ejemplo los carecteres gráficos. Su uso queda reservado exclusivamente para tu ordenador; los ordenadores de otras marcas, no disponiendo del correspondiente carácter, no están en condiciones de reconocerlos y en consecuencia de usarlos.
Es bueno que tengas presente esta limitación si desea escribir programas «portátiles», es decir, que puedan funcionar también en otras máquinas.

CODE

A veces puede resultar útil conocer el código ASCII de un carácter determinado, o al contrario, producir e imprimir un carácter a partir de su código. El BASIC pone a tu disposición dos sentencias con las que puedes convertir los caracteres en códigos y los códigos en caracteres, son: CODE y CHR$. Ambas son funciones, y en consecuencia será necesario que a cada una de ellas le proporciones el argumento sobre el que operar.



CODE produce el valor del código ASCII, es decir, un número, correspondiente al primer carácter de una cadena.
Por lo tanto, el argumento deberá ser una cadena, bajo forma de constante (encerrada naturalmente entre comillas), o una variable.
CODE proporciona como resultado un número comprendido entre 0 y 255, puesto que los caracteres disponibles en tu ordenador, como ya hemos visto, son en total 256.
Si el argumento fuera una cadena vacía, CODE devuelve el valor 0.
Veamos algunos ejemplos:

```
PRINT CODE («A»)
```

Obtendrás 65, el código numérico ASCII del carácter «A».

Sintaxis de la instrucción

CODE (cadena)

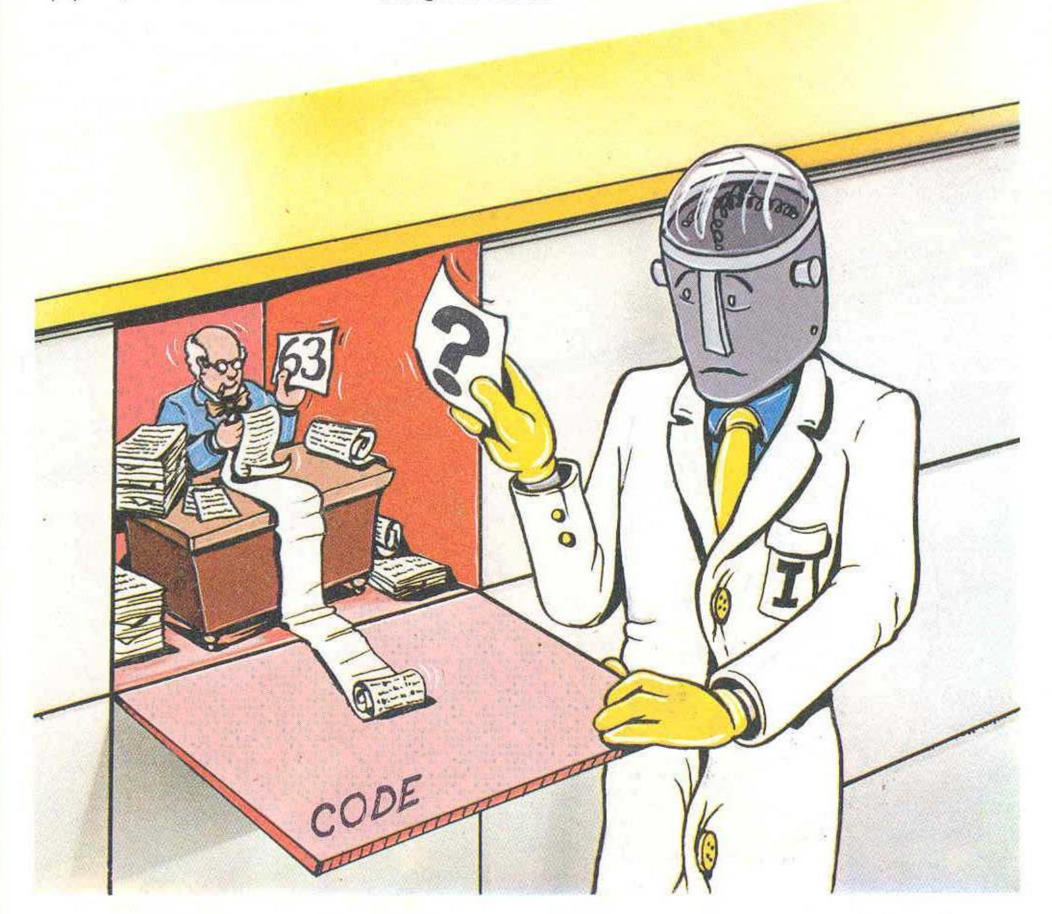
```
PRINT CODE («ABCD»)
```

Obtendrás 65: el primer carácter del argumento es «A».

```
50 C$ = «TABLA»
60 PRINT CODE (C$)
```

Obtendrás 84, código numérico del carácter "T".

El argumento de CODE es un carácter. El resultado es su código numérico.



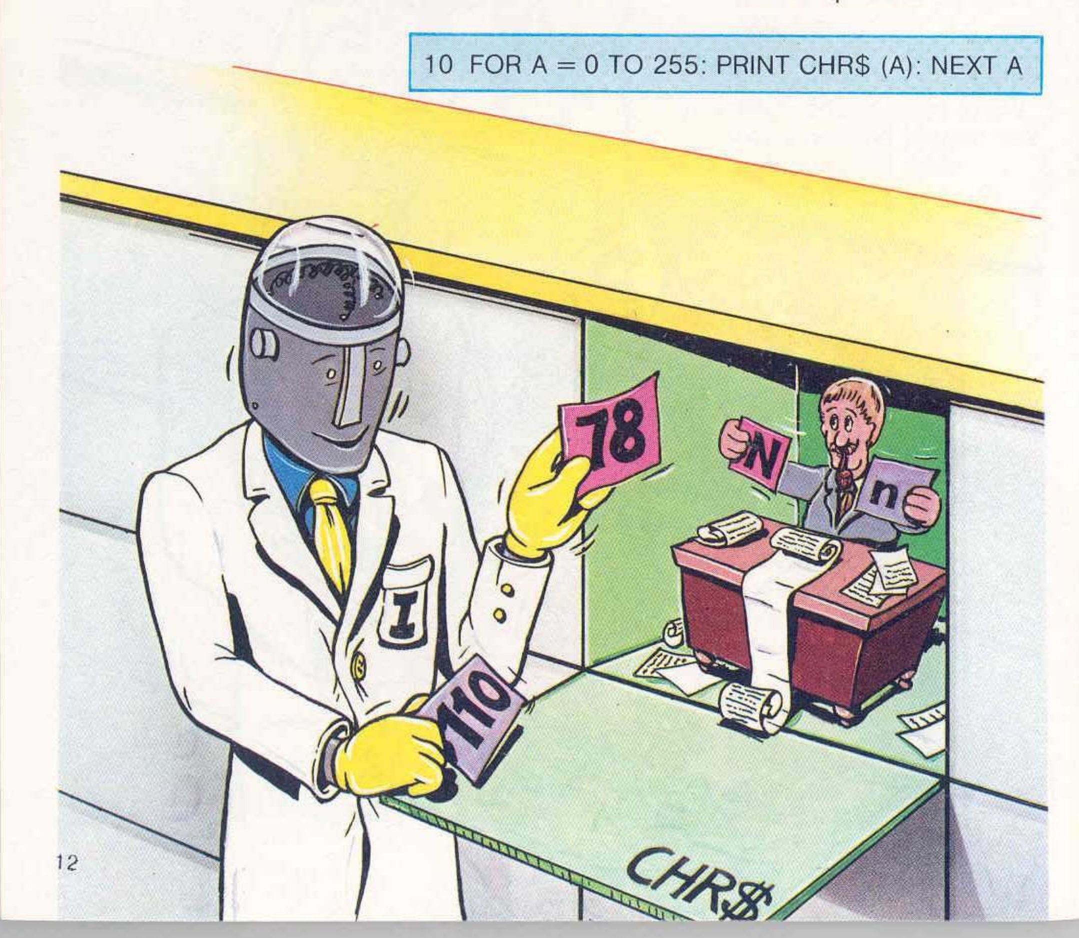
CHR$

CHR$ es, en cierto sentido, la función opuesta de CODE: te devuelve el carácter correspondiente al número que hayas empleado como argumento.
Este número se puede especificar mediante una variable o una expresión.
El argumento deberá quedar comprendido entre 0 y 255; en caso contrario, el ordenador visualizará el mensaje de error INTEGER OUT OF RANGE.
Cuando el valor del argumento sea un número decimal, CHR$ lo redondea al valor entero obtenido sumando 0,5.

```
15 PRINT CHR$(151/2)
```

151/2 = 75.5: el número entero que se obtiene al sumar 0.5 es 76. A este argumento le corresponderá el carácter L.
El siguiente programa imprime el juego completo de caracteres de tu Spectrum:

```
10 FOR A = 0 TO 255: PRINT CHR$ (A): NEXT A
```



Algunos de estos caracteres (los llamados caracteres de control), aún formando parte del juego de caracteres, no son visualizables; por lo tanto, no aparecerán en pantalla.
Las funciones CODE y CHR$ resultan útiles si se desea transformar una letra mayúscula en su correspondiente carácter en minúscula, y viceversa.
Estudiando la tabla ASCII podrás darte cuenta rápidamente de que el código de una letra minúscula es igual al código de su mayúscula correspondiente sumándole 32.
El programa siguiente, aprovecha precisamente esta característica para imprimir en minúscula el carácter correspondiente a la mayúscula que se haya tecleado.

El argumento de CHR$ es un código numérico cuyo resultado es el carácter correspondiente.

```
10 CLS
20 INPUT A$: IF A$ = «*» THEN GOTO 90: REM
   espera la pulsación de una tecla.
30 IF CODE (A$) < 65 THEN GOTO 20.
40 IF CODE (A$) > 90 THEN IF CODE (A$) < 97
   THEN GOTO 20
50 IF CODE (A$) > 122 THEN GOTO: REM esta
   concatenación de IF hace que se acepten
   únicamente los caracteres correspondientes a
   las letras mayúsculas, entre 65 y 90 y de las
   minúsculas, entre 97 y 122, ignorándose todos
   los demás.
60 IF CODE (A$) < 91 THEN PRINT A$, CHR$
   (CODE (A$) + 32) : REM si el carácter está en
   mayúsculas, entonces imprime también el
   correspondiente carácter en minúscula.
70 IF CODE (A$) > 96 THEN PRINT A$, CHR$
   (CODE (A$) – 32) : REM si, por el contrario,
   fuera minúscula, entonces imprime también la
   correspondiente mayúscula.
80 GOTO 20
90 REM FIN
```

Ejemplos

```
PRINT CHR$ (65)
```

El código 65 corresponde al carácter «A».

```
PRINT CHR$ (18 * 5)
```

El argumento de la función CHR$ puede ser una expresión cuyo valor esté comprendido entre 0 y 255.

```
S = (100 – 1)
PRINT CHR$ (S/3)
```

Si miras en la tabla ASCII publicada algunas paginas atrás, verás que el código 33 corresponde a la admiración (!).

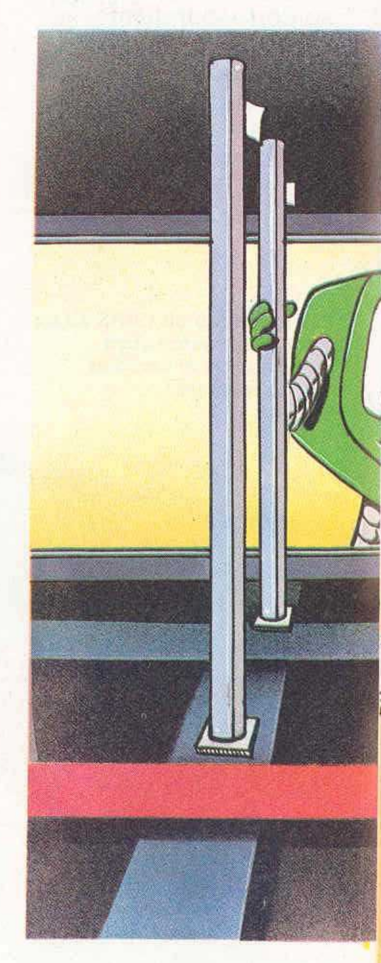
```
PRINT CHR$ (65.91)
```

De acuerdo con lo dicho anteriormente, el valor entero que se obtiene sumando 0.5 es 66.
Se obtiene por lo tanto la impresión del carácter B.

Sintaxis de la instrucción

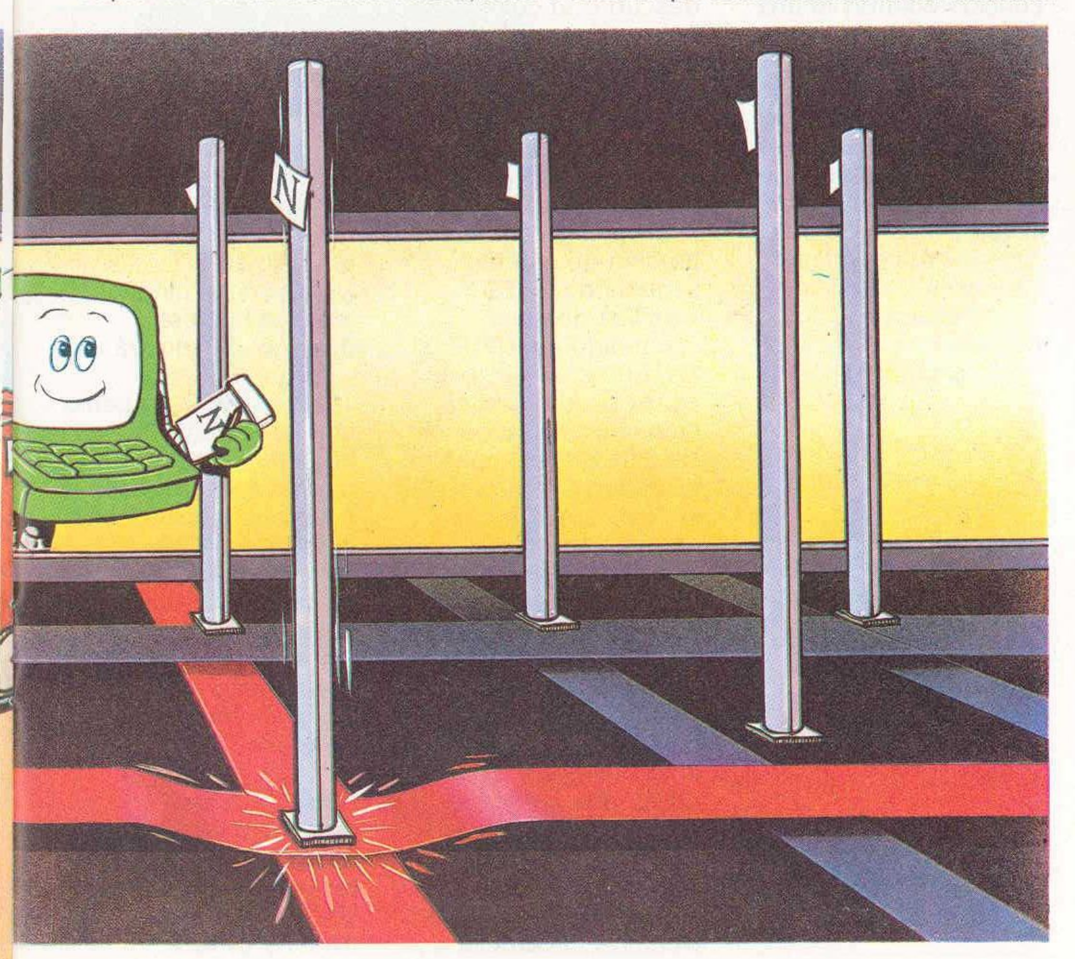
CHR$ (número)
Donde NUMERO puede ser un número, una variable o una expresión numérica

INKEY$



La función INKEY$ permite la entrada desde el teclado de tu Spectrum de un único carácter, pero sin tener que pulsar después la tecla ENTER. Además el carácter correspondiente a la tecla pulsada no se visualiza en pantalla.
¿Por qué no emplear entonces el conocido y familiar INPUT?
¿No es también cierto que con INPUT también se puede introducir un único carácter? La diferencia es sutil, pero importante.
Como ya se ha visto hablando de INPUT, si te equivocas en el dato a introducir, las consecuencias pueden ser catastróficas, los resultados equivocados, o aún peor, se puede bloquear el programa. Con INKEY$ no ocurre

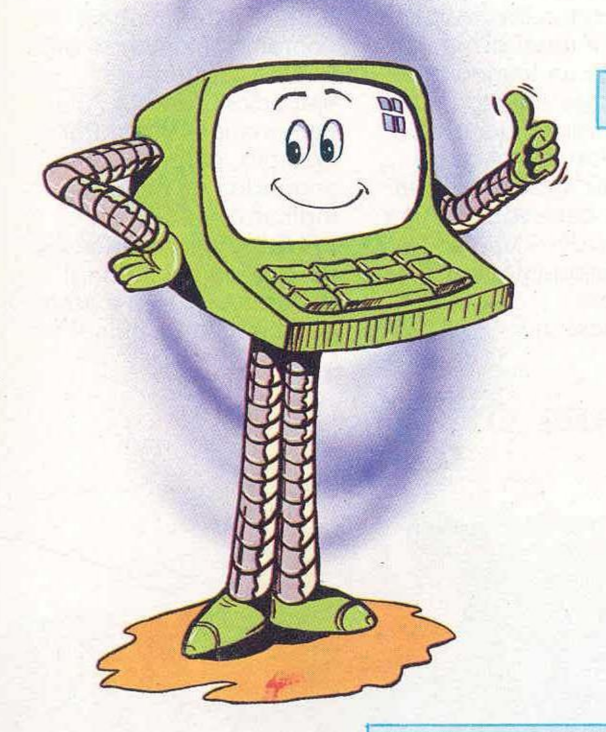
Tu Spectrum cuando encuentra INKEY$, anota el carácter que estés tecleando en ese momento.



nada de todo esto: parece hecho aposta para que puedas equivocarte (INPUT controlado) sin que ocurra nada. Tu Spectrum espera tranquilamente que pulses la tecla correcta. El término «espera» merece unas consideraciones aparte. Con INPUT, el programa se interrumpe para esperar los datos de entrada, en cambio, INKEY$ se ejecuta como una función normal del BASIC, con la velocidad de la que es capaz el intérprete, muy superior a la de cualquier ser humano. Por lo tanto, será necesario insertar la función INKEY$ dentro de un bucle de espera, que haga más «humano» el tiempo de respuesta.
Si quieres puedes descubrir tú solo (naturalmente sin leer el listado) cuándo los programas, por ejemplo los de VIDEOBASIC, emplean la función INKEY$ o un INPUT. Piénsalo... cada vez que el programa requiera la presión de una tecla seguida de ENTER significa que hay escondido un INPUT; si ENTER no es necesaria se tratará de INKEY$, cuyo argumento puede ser una o más variables de cadena. Lógicamente, si deseas que el carácter reconocido sea 2, o *, o bien el !, será imprescindible que lo coloques entre comillas.
En resumen, INKEY$ se emplea habitualmente para esperar y comprobar la entrada de un carácter cualquiera desde el teclado, o para esperar a que el usuario pulse una tecla.
La instrucción INKEY$ devuelve el carácter tecleado tan pronto como se recurre a ella. Cuando no esté pulsada (o no haya sido pulsada) ninguna tecla, INKEY$ asigna a la variable un valor nulo, y el programa continúa con toda normalidad.
El ejemplo siguiente te aclarará las ideas:

```
10 LET A$ = INKEY$
20 IF A$=" " THEN 10
30 PRINT A$
```

La línea 10 asigna a la variable A$ el valor de la tecla que hayas pulsado en tu Spectrum; si ninguna tecla se hubiera pulsado, A$ tomará un valor nulo.
Tanto en un caso como en el otro, el programa no sufrirá ninguna parada o interrupción.
La línea 20 comprueba el valor de A$: si el valor fuera nulo, la ejecución volverá a la línea 10, volviendo a empezar el ciclo.
Por lo tanto, la única forma de terminar el programa será pulsar un tecla cualquiera. Haciendo esto, aparecerá además en pantalla el carácter tecleado (línea 30).
Un uso típico de INKEY$ es fácil



encontrarlo en aquellos programas que someten al usuario a elecciones de este tipo:

¿QUIERES CONTINUAR? (S/N)

Estas preguntas requieren como respuesta la pulsación de las teclas S o N, construyéndose una estructura ciclica semejante a la anteriormente explicada, siendo así posible descartar automáticamente todas las contestaciones que no estén entre las admisibles, y evitando que quizá pudiera llenarse la pantalla con caracteres inútiles y antiestéticos.

```
10 CLS
20 PRINT «¿QUIERES CONTINUAR? (S/N)»
30 LET R$ = INKEY$
40 IF R$ <> «S» THEN IF R$ <> «N» THEN
   GOTO 30
50 PRINT R$
60 IF R$ = «S» THEN GOTO 20
70 REM FIN
```

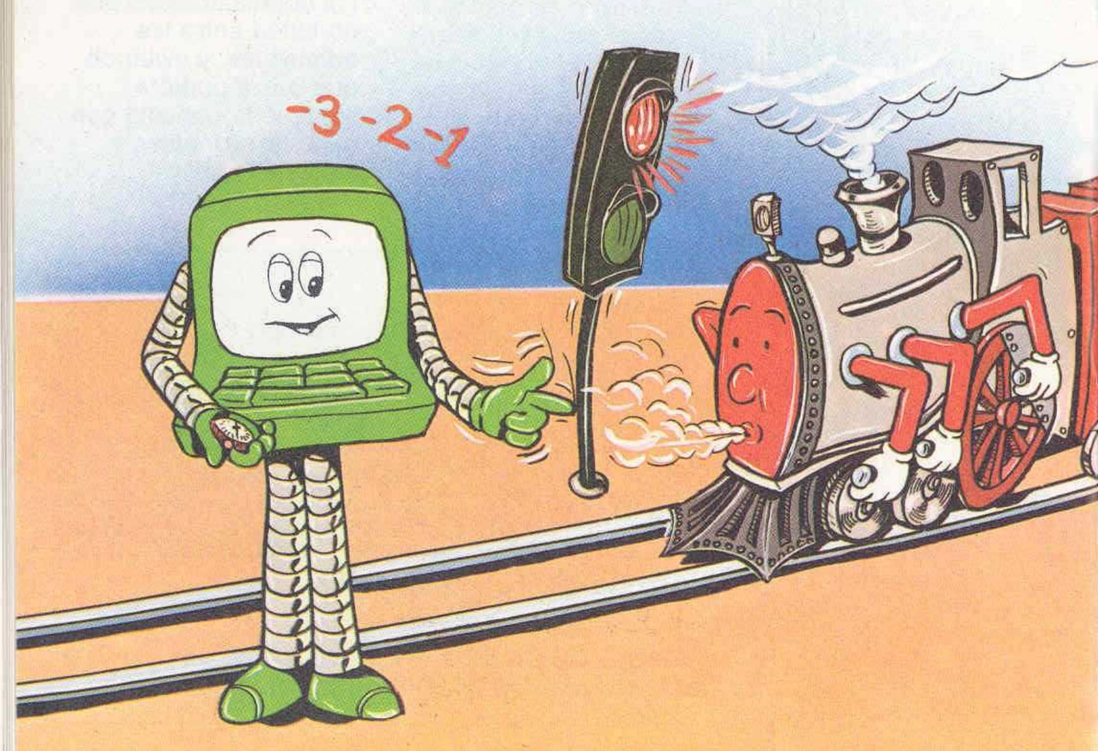
Sintaxis de la instrucción

VARIABLE DE CADENA = INKEY$
Como observarás, INKEY$ carece de argumento.

PAUSE

A veces, durante la ejecución de un programa, puede resultar útil tener la posibilidad de parar momentáneamente la ejecución de las instrucciones.
En algunos casos se hacen necesarias pausas que le permitan al usuario leer o valorar una determinada visualización, o tomar cualquier decisión.

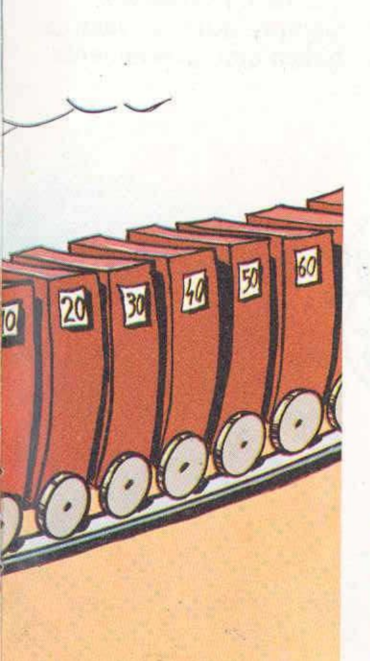
Con este objeto, tu Sinclair dispone de una orden especial: PAUSE.
PAUSE n, para la ejecución del programa durante un tiempo tanto más largo cuanto mayor sea el valor numérico asignado a n, y como máximo 65535, valor al que le corresponde una parada de aproximadamente 22 minutos.
Si indicas n = 0, el programa parará mientras no pulses una tecla.
Valores intermedios, comprendidos entre 0 y 65535 pararán la ejecución durante tiempos inferiores. Por ejemplo, para n = 50 segundos es necesario indicar n = 250.
PAUSE dispone además de otra particularidad: si durante el transcurso de la pausa se pulsa



una tecla cualquiera, el programa empieza de nuevo inmediatamente a trabajar, acortando la pausa originalmente prevista.
Así ésta instrucción se puede usar para adecuar el tiempo de permanencia en pantalla de las distintas visualizaciones, a la capacidad de lectura del usuario del programa.

Sintaxis de la instrucción

PAUSA n
Donde n es un valor numérico comprendido entre 0 y 65535.



FOR, TO, STEP, NEXT

Ocurre con frecuencia que en un programa sea necesario repetir una instrucción o grupo de instrucciones.
Supón, a título de ejemplo, que deseas escribir un programa que multiplique la variable A por los valores 1, 2, 3, 4 y 5.
Una posible solución sería ésta:

```
10 LET I = 1 : REM I
   es el número a
   multiplicar por A.
20 LET A = A * I
30 LET I = I + 1
40 IF I < 6 GOTO 20
```

Se trata de un ejemplo típico de bucle, es decir una secuencia de instrucciones ejecutadas una cierta cantidad de veces.
Existe en BASIC una instrucción especial que te permite realizar estos bucles sin recurrir a estructuras complejas.
La instrucción está compuesta de las palabras FOR...NEXT.
Imagínate que desees repetir una serie de instrucciones un número de veces determinado, y exactamente hasta que un contador C (cuyo valor inicial es S) alcance el valor A.
Empleando la instrucción FOR...NEXT podrás escribir:

```
FOR C = S TO A
```

haz alguna cosa.

```
NEXT C
```

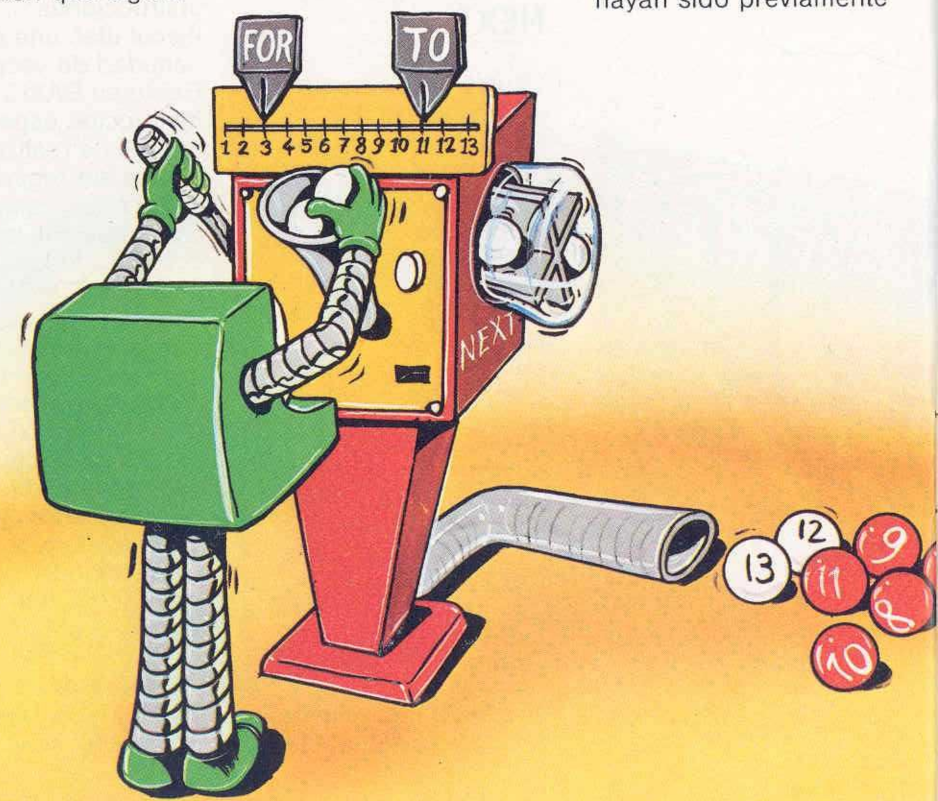
La primera vez que se ejecuta el bucle, C se establece con un valor igual a S. Luego se ejecutan todas las instrucciones.
Al llegar al NEXT, el valor de C es incrementado y comparado automáticamente con A. Si C resultara menor que A, el ciclo se repite, si no se continúa con la instrucción que siga al NEXT.
Por lo tanto, el procedimiento se repite hasta que el contador C alcanza el valor de A.
Viendo de nuevo el problema de la multiplicación, podría dársele entonces esta otra solución:

```
10 FOR C = 1 TO 5
20 LET A = A * C
30 PRINT A
40 NEXT C
```

C se llama variable de control del ciclo (o contador) y puede tomar cualquier nombre legal permitido.
Habrás notado en el ejemplo que 1 es su valor inicial y 5 el valor final o de test.
En la instrucción FOR también se pueden usar expresiones, por ejemplo:

```
FOR A = M + 5 TO B/5
```

y también variables, siempre que sus valores hayan sido previamente



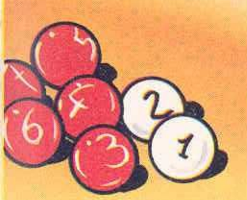
establecidos. Por ejemplo:

```
10 LET DA = 5 : LET A = 15
20 FOR C = DA TO A
30 ...
40 NEXT C
```

Además es posible incrementar el valor de la variable de contador con un paso distinto de 1, utilizando simplemente la palabra STEP (paso) seguida del valor con que se desee incrementar cada vez el contador:

```
FOR I = 2 TO 10 STEP 2
```

I asumirá entonces los valores 2, 4, 6, 8, 10. También es posible escribir programas en los cuales los ciclos contengan en su interior otros ciclos:



```
10 FOR I=0 TO 10
   STEP 3
20 FOR J=3 TO 9
30 ...
40 ...
50 ...
60 NEXT J
70 NEXT I
```

Se dice entonces que los ciclos están concatenados (acuérdate de IF...THEN...GOTO). Naturalmente, los contadores de los distintos ciclos deberán ser distintos. El ciclo comprendido en otro ciclo deberá además quedar totalmente contenido en el primero; por lo tanto, no es posible superponer partes de ciclos:

```
100 FOR I=13 TO 20
110 FOR J=0 TO 10
120 ...
130 ...
140 ...
150 ...
160 NEXT I
170 NEXT J
```

¡Está equivocado! Los dos ciclos están superpuestos. ¡Es muy peligroso! Si la variable de indice es inicialmente mayor que el valor final, el ciclo se ejecutará una sola vez. Por ejemplo:

```
50 FOR I = 20 TO 5
60 PRINT I
70 NEXT I
```

será prácticamente ignorado.
El paso del ciclo también puede ser negativo:

```
30 FOR X = 13 TO 10 STEP -2
40 ...
50 ...
60 ...
70 NEXT X
```

En este caso el bucle se ejecuta hasta que X, decrementándose en 2 cada vez, se haga menor que 10.
Si el valor del paso se establece como igual a 0, el ciclo se repite indefinidamente.

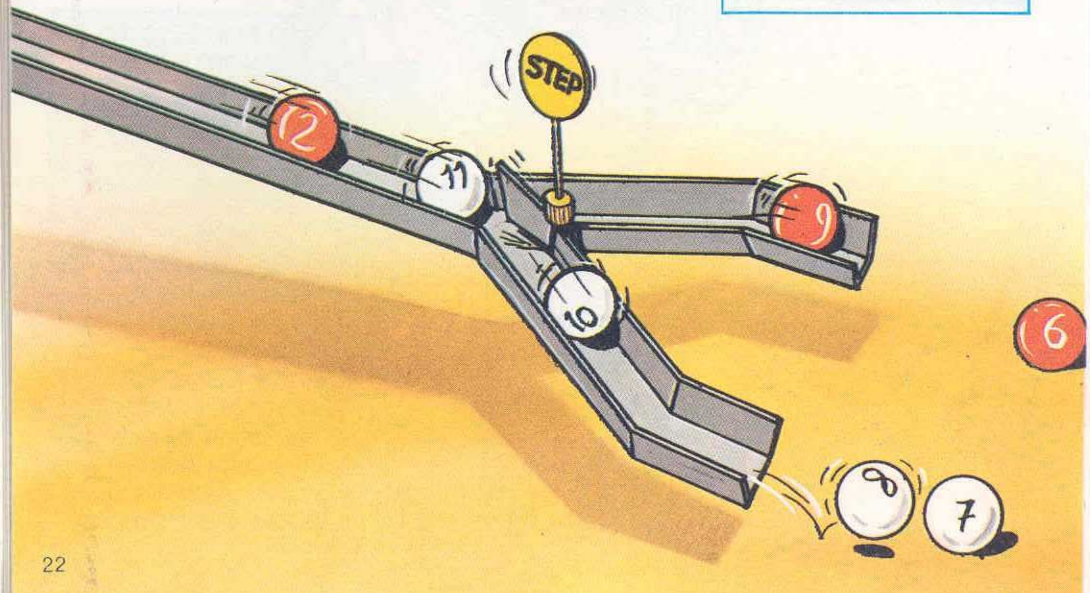
```
10 FOR K = 1 TO 10 STEP 0
20 PRINT K
30 NEXT K
```

El único resultado de este programa será, por lo tanto, la visualización continua en pantalla del valor inicial de K, es decir, 0.
Además, deberás prestar atención al hecho de que salir desde el interior de un bucle antes de que la variable de control haya alcanzado el valor final, hará que el ordenador espere el cierre de un ciclo que no encontrará nunca. En algunos casos esto podría llegar a provocar mensajes de este tipo:

```
NEXT WITHOUT FOR
```

o bien

```
VARIABLE NOT FOUND
```



Por lo tanto, es una práctica aconsejable el evitar incluir en un bucle instrucciones de salto (es decir GOTO); esto será también ventajoso para la legibilidad del programa.
Otro error muy común es el de emplear un número de NEXT superior al de los FOR. En este caso tu Spectrum contestará con:

```
NEXT WITHOUT FOR
```

Una última recomendación: el nombre de la variable de control deberá ser siempre una sola letra, como i, j, k, x...

Sintaxis de la instrucción

FOR indice = valor inicial TO valor final [STEP paso]
NEXT indice

En la cual:

- Indice es el nombre de una variable numérica empleada como contador.
- Valor inicial es el valor de salida del indice.
- Valor final es el valor que el indice tiene que igualar o superar.
- Paso es el incremento que experimenta el indice con cada repetición. Si no ha sido especificado se establece como 1. Puede ser negativo.



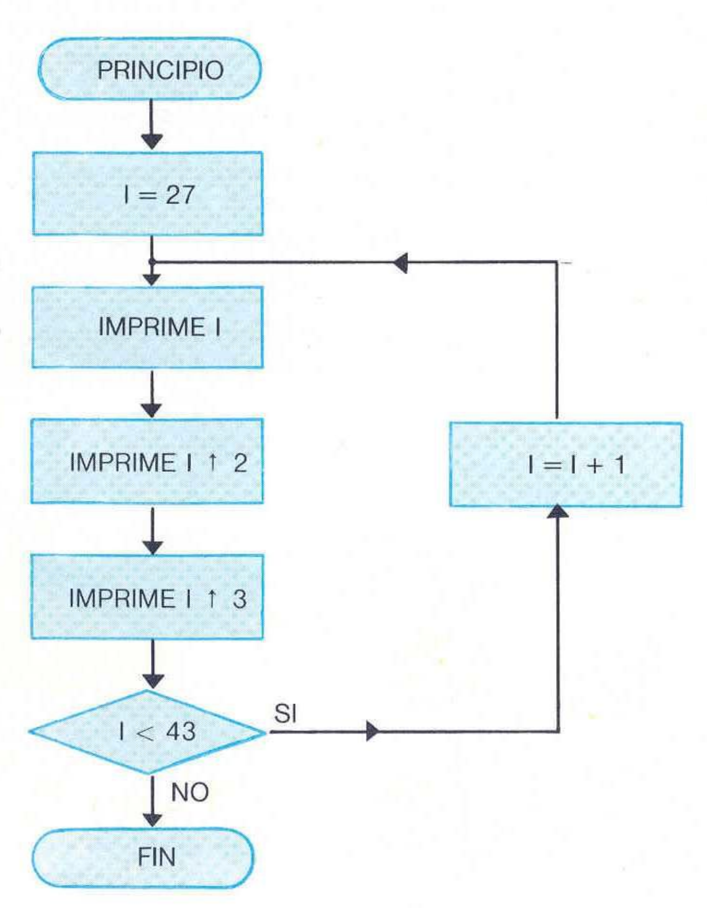


Los ciclos automáticos

Llamamos ciclos automáticos a todos aquellos ciclos que ejecutan repetidamente una secuencia de instrucciones, empleando la instrucción FOR...NEXT.
En la mayor parte de los programas, los ciclos automáticos son tan comunes y útiles que constituyen un instrumento importantísimo en manos de cualquier programador.
Los ejemplos de aplicaciones que vienen a continuación, te ayudarán a aclarar y a profundizar los conceptos teóricos que ya conoces, tanto respecto del uso de ciclos como de la instrucción FOR...NEXT.

Cuadrados y cubos

Como primer ejemplo estudiaremos este problema: encontrar los cuadrados y los cubos de los números comprendidos entre 27 y 43.
Una posible solución podría ser:



A este diagrama de flujo le corresponden los dos siguientes programas BASIC.
El primero emplea un ciclo controlado (es decir, creado «artificialmente» por el programador); el segundo usa un ciclo automático:

```
10 LET I = 27
20 PRINT I ↑ 2,
30 PRINT I ↑ 3
40 IF I<43 THEN I = I+1:GOTO20
50 REM FIN
```

```
10 FOR I=27 TO 43
20 PRINT I ↑ 2,
30 PRINT I ↑ 3
40 NEXT I
50 REM FIN
```

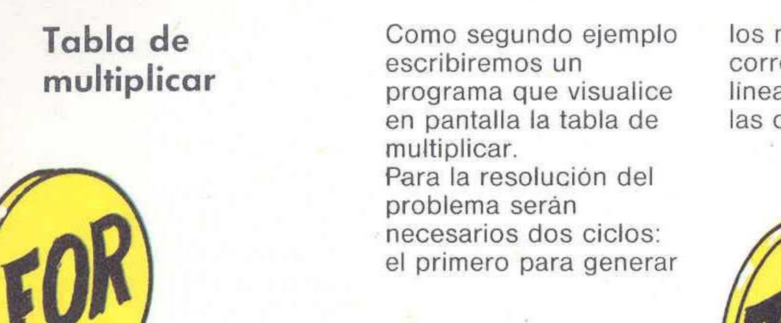
Saltará inmediatamente a la vista del lector, la mayor legibilidad del programa con el ciclo automático; se distinguen en seguida en él el principio y el final del ciclo, lo que no ocurre en el primer listado.

También para tu Spectrum es preferible la segunda solución, puesto que la ejecución del ciclo queda confiada a una instrucción concebida especialmente para resolver este tipo de problemas, y por lo tanto específica para ellos.

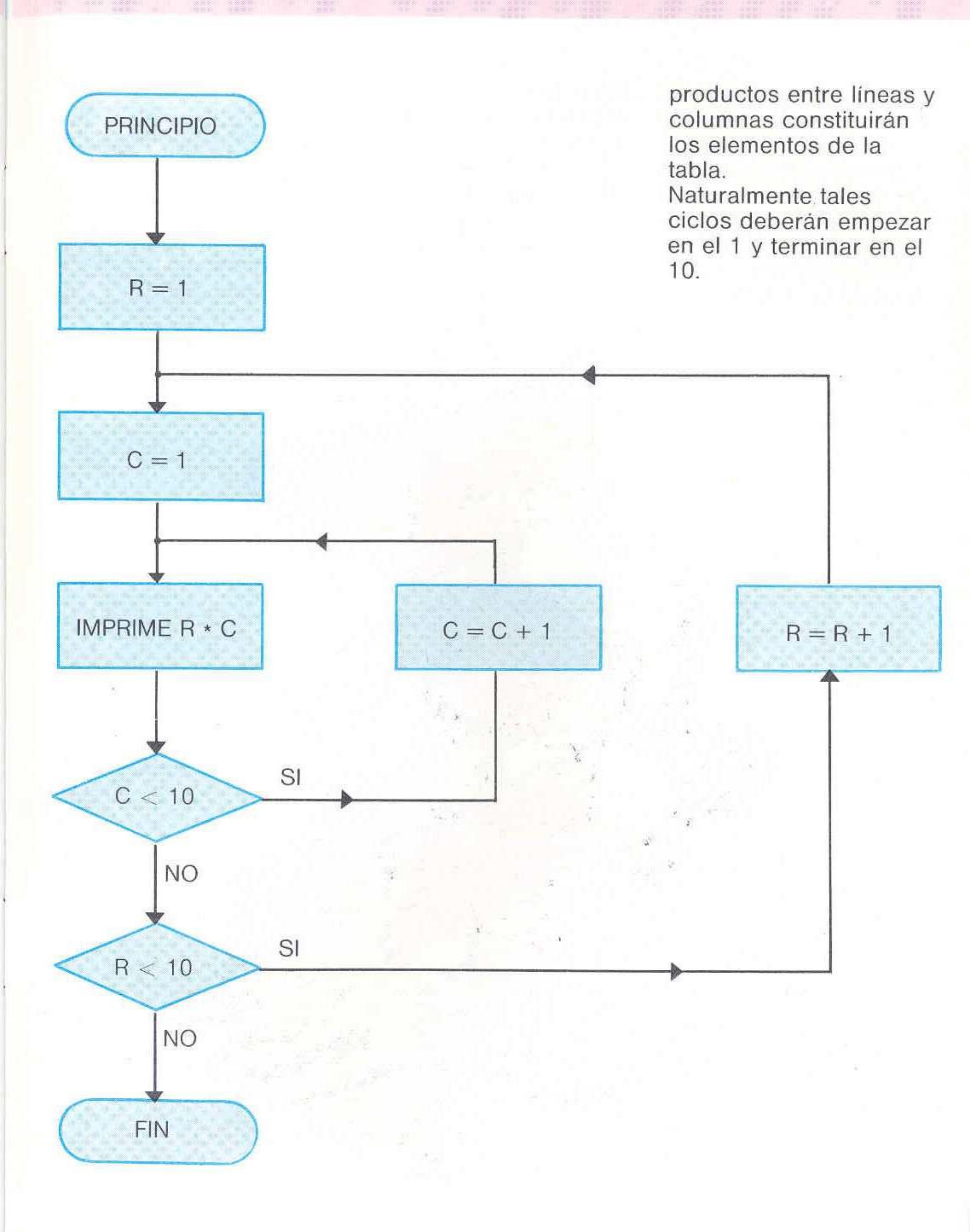
En los casos en que sea posible (y son la mayoría) será siempre mejor recurrir a un ciclo automático: el programa —y el programador— resultarán beneficiados desde todos los puntos de vista.

Tabla de multiplicar

Como segundo ejemplo escribiremos un programa que visualice en pantalla la tabla de multiplicar.
Para la resolución del problema serán necesarios dos ciclos: el primero para generar los números correspondientes a las líneas, el segundo para las columnas. Los

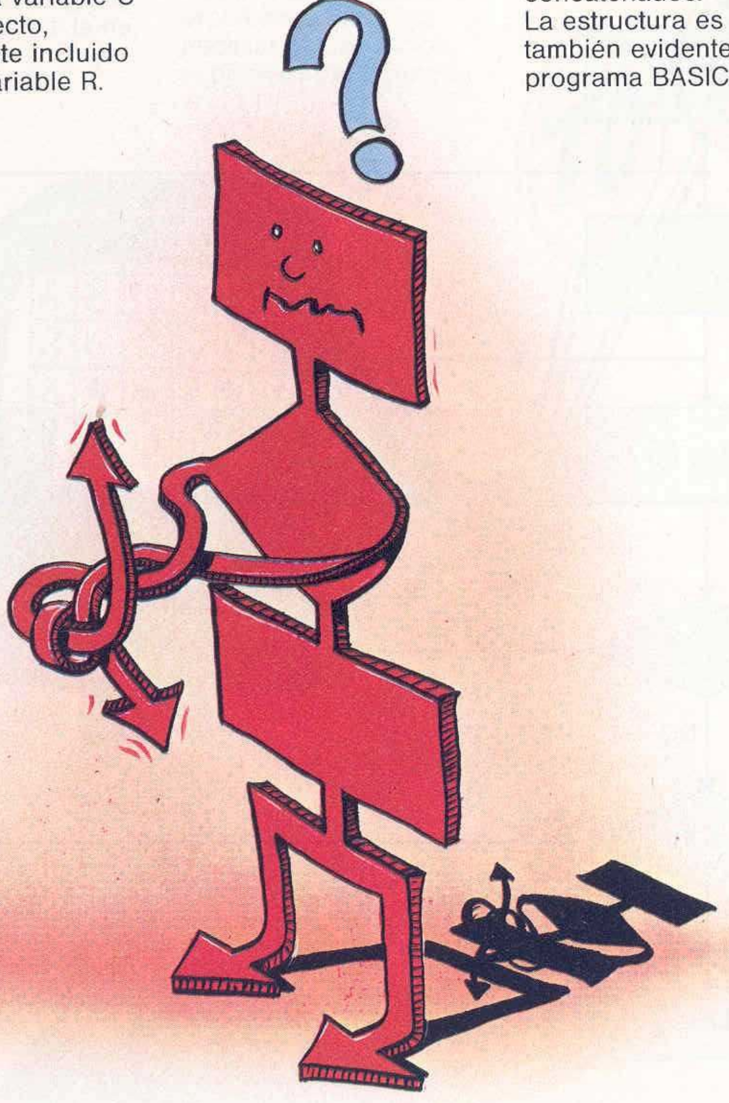


productos entre líneas y columnas constituirán los elementos de la tabla.
Naturalmente tales ciclos deberán empezar en el 1 y terminar en el 10.



El diagrama de flujo ilustra perfectamente el concepto de «bucles concatenados»; el ciclo que modifica e incrementa la variable C queda, en efecto, completamente incluido en el de la variable R.

En tu Spectrum los ciclos FOR se pueden concatenar, es decir, situarse los unos dentro de los otros, hasta un máximo de 10 veces. Pero no abuses, porque se hace muy difícil seguir el flujo de más de tres ciclos concatenados.
La estructura es también evidente en el programa BASIC:



```
10 FOR R = 1 TO 10
20 FOR C = 1 TO 10
30 PRINT R * C; " ";
40 IF R * C < THEN PRINT " ";: REM si el
   producto está compuesto por una sola cifra
   entonces imprime un espacio.
50 NEXT C
60 PRINT
70 NEXT R
80 REM FIN
```

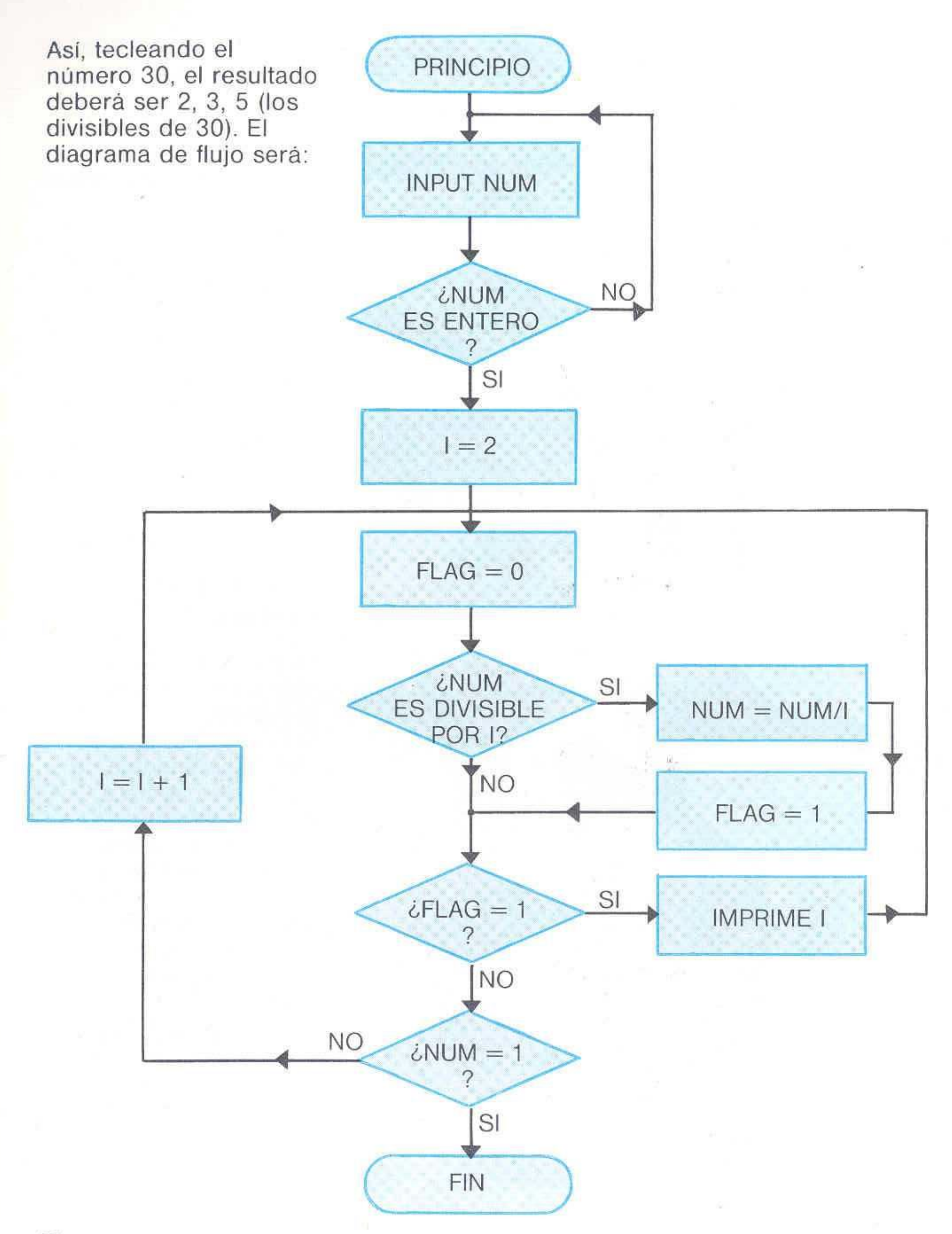


Las líneas 40 y 60 han sido incluidas en el programa para alinear las tablas, incluyendo los espacios oportunos entre las distintas líneas y columnas.
Para comprender cuál es su efecto, prueba a eliminarlas (comienza por la línea 40 y después con la 60): por lo que aparezca en pantalla deberías ser capaz de comprender inmediatamente su función y eficacia.

Descomposición en factores primos

Como último ejemplo veamos el siguiente problema.
Escribir un programa que, aceptando como entrada un número entero cualquiera, produzca en la salida todos los valores de este número que sean sus factores primos (recuerda que los números primos son aquellos números únicamente divisibles por 1 y por sí mismos).

Así, tecleando el número 30, el resultado deberá ser 2, 3, 5 (los divisibles de 30). El diagrama de flujo será:



He aquí el correspondiente listado BASIC:

```
 5 INPUT "NUMERO = " ; NUM
10 CLS: PRINT "FACTORES PRIMOS DE" ; NUM
20 IF NUM <>INT (NUM) GOTO 5: REM ¿NUM ES UN NUMERO ENTERO?
30 FOR I = 2 TO NUM
40 LET FLAG = 0:REM FLAG <>0 CUANDO NUM ES DIVISIBLE POR I
50 IF NUM/I = INT (NUM/I) THEN LET NUM = NUM/I : LET FLAG = 1
60 IF FLAG = 1 THEN PRINT I: GOTO 40
70 IF NUM = 1 THEN GOTO 90
80 NEXT I
90 REM FIN
```

El programa empieza comprobando que el valor del número tecleado no tenga cifras decimales; en el caso contrario solicita la entrada de un nuevo número.
Desde la línea 30 empieza la auténtica fase de ejecución y resolución del problema: si NUM (el número tecleado como dato) es divisible por I (línea 50), entonces FLAG toma el valor 1. FLAG es una variable empleada como indicador: cuando toma el valor 1 significa que I es un divisor de NUM y por lo tanto debe imprimirse. Si en cambio vale 0, I no es divisor de NUM, y por lo tanto hay que incrementarle en 1. A medida que el ciclo continúa, NUM se hace sucesivamente más pequeño; al final tomará el valor 1.
En este momento el problema habrá sido resuelto: habrán aparecido en pantalla todos aquellos números que multiplicados entre ellos, formaban el valor inicial de NUM.

EJERCICIOS

Ayudándote con la función CODE dispón en orden creciente, según los códigos, las siguientes cadenas:

"RUN"

"TU MICRO"

"SOFTWARE",

CHR$ (13) + "VIDEOJ".

"VIDEOBASIC"

CODIGO	CADENA

Ayudándote con la tabla ASCII determina y escribe la salida del siguiente programa:

```
10 PRINT CHR$ (86); CHR$ (73); CHR$ (68); CHR$ (69); CHR$ (79);
20 PRINT CHR$ (66); CHR$ (65);
30 PRINT CHR$ (83); CHR$ (73); CHR$ (67)
```


A) Cronometra y anota la duración del programa.
B) Busca, anota y después elimina la línea de retardo.
C) Cronometra nuevamente y anota el tiempo.

```
10 CLS
20 FOR D = 10 TO 90 STEP 8
30 PRINT "DATO"; D
40 FOR P = 1 TO 3000: NEXT P
50 NEXT D
```

A	TIEMPO
B	N.º DE LA LINEA DE RETARDO
C	TIEMPO

Sustituye ahora la línea 40 del anterior programa con la siguiente:
40 PAUSE 150
Cronometra nuevamente y busca un valor de pausa análogo al obtenido con FOR P = 1 TO 3000: NEXT P.



El fruto de la Investigación

La nueva impresora de **SEIKOSHA SP-800,** con un ordenador personal puede escribir ***96 combinaciones de letra diferentes,*** desde 96 caracteres por segundo a 20 con muy alta calidad de letra, además es gráfica en alta densidad.

Su precio es de 69.900 Pts con ***introductor automático hoja a hoja.***

Con un pequeño ordenador personal, un procesador de textos puede costar alrededor de cien mil pesetas.

Infórmese y comprenderá por qué **las máquinas de escribir** tienen demasiados años.

Nuestra calidad es "SEIKO";
nuestros precios, únicos
Si desea más información,
consulte con nuestro distribuidor
más cercano, llame o escriba a:



DIRECCION COMERCIAL:
Av. Blasco Ibañez, 114-116
46022 VALENCIA
Tel (96) 372 88 89
Télex 62220
DIRECCION COMERCIAL EN CATALUÑA:
C/Muntaner, 60-2-4Pta
08011 BARCELONA
Tel (93) 323 32 19

ESTOS SON NUESTROS MODELOS:

MODELO	VELOCIDAD	COLUMNAS	TIPOS DE LETRA	P.V.P.R. INTERFACE PARALELO
GP-50S LA DEL SPECTRUM	40 cps	32	-	19.900
GP-50 LA PEQUEÑA	40 cps	46	2	25.900
GP-500 LA ECONOMICA	50 cps	80	2	47.900
GP-700 LA DE COLOR	50 cps	80-106	3	69.900
SP-800 LA PERFECCION	96 cps	80-137	20	69.900
BP-5200 LA DE OFICINA	200 cps	136-272	18	199.900
BP-5420 LA MAS RAPIDA	420 cps	136-272	18	299.900

* Los precios indicados son los recomendados para conexion tipo paralelo Centronics, para otro tipo de conexion, sufren un ligero incremento.

Este pie de página ha sido realizado integramente con la nueva impresora:

SEIKOSHA SP-800